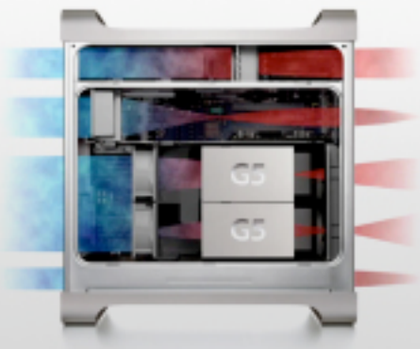
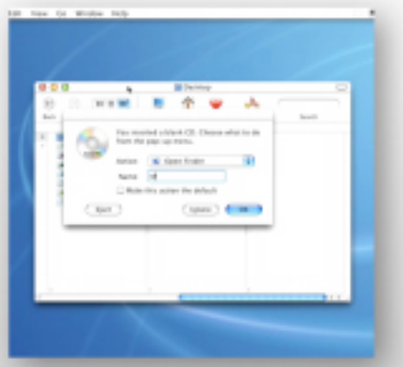
Mac OS X High Performance Math Libraries

Session 507



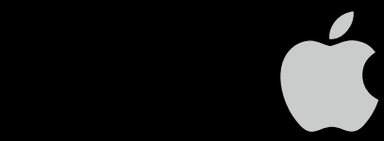


Mac OS X High Performance Math Libraries

Steve Peters, Ph.D.
Vector and Numerics Group
scp@apple.com

Introduction

- We want your apps to achieve optimum floating point performance!
- We want your apps to deliver robust, high quality numerical results
- We want your apps to readily port to the industry standard math APIs in the MacOS X frameworks
- We want your apps to enjoy easy access to Apple value-added features

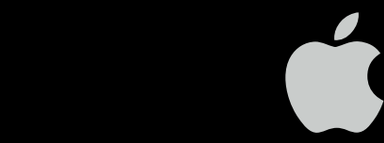


What You Will Learn

- We will inform you about the floating point performance work Apple is delivering
- We will show you how to leverage that work in your development
- We will walk you through a regimen you can follow to gain floating point speed-ups
- We will recommend some learning resources

Technology Framework

- The silicon beneath your feet:
  - PowerPC double precision Floating Point Core(s), implements IEEE-754 compliant operations
  - AltiVec Java compliant single precision vector engine [SIMD unit]
- The Frameworks you can build on:
  - libm (a sub library of libSystem, always linked)
  - vecLib and vImage, part of the NEW Accelerate.framework
- The Developer Tools are your best friends:
  - CHUD Toolkit



Delivered In Jaguar: Libm

- Libm Design Center:
  - Standards conforming API: IEEE754, C99 (for double)
  - Numerically robust (core approximations correctly rounded for all arguments, others at 1ulp)
  - Best of breed algorithms

- Coding to Libm in C:

```
#include <math.h>
#include <fenv.h>                    // for special needs

cc -o myMath myMath.c
```

Aside: Can I Go Faster Than Libm?

Aside: Can I Go Faster Than Libm?

- Well, yes. Because:
  - Libm emphasizes numerical robustness
  - Standard APIs data-starve the FPU
  - Libm is obliged to handle NaNs, Infinities, Zeroes
  - Libm preserves, protects, and updates FP status flags

Aside: Can I Go Faster Than Libm?

- Well, yes. Because:
  - Libm emphasizes numerical robustness
  - Standard APIs data-starve the FPU
  - Libm is obliged to handle NaNs, Infinities, Zeroes
  - Libm preserves, protects, and updates FP status flags

```
double fast_hypot ( double x, double y )
{
    return sqrt( (x * x) + (y * y) );
}

fast_hypot( 3.0, 4.0 ) -> 5.0
fast_hypot( 3.0e160, 4.0e160 ) -> INF
```

```
double hypot ( double x, double y )
{
        register double temp;
        hexdouble OldEnvironment, CurrentEnvironment;

        register double FPR_env, FPR_z, FPR_one, FPR_inf, FPR_Minf,
                        FPR_absx, FPR_absy, FPR_big, FPR_small;

        FPR_z = 0.0f;                                   FPR_one = 1.0f;
        FPR_inf = Huge.d;                               FPR_Minf = NegHuge.d;
        FPR_absx = __FABS( x );                         FPR_absy = __FABS( y );

        if ( ( x == FPR_inf ) || ( y == FPR_inf ) || ( x == FPR_Minf ) || ( y == FPR_Minf ) )
            return FPR_inf;

        if ( ( x != x ) || ( y != y ) ) {
            x = __FABS ( x + y );
            return x;
        }
        if ( FPR_absx > FPR_absy ) {
            FPR_big = FPR_absx;
            FPR_small = FPR_absy;
        } else {
            FPR_big = FPR_absy;
            FPR_small = FPR_absx;
        }

        // Now +0.0 <= FPR_small <= FPR_big < INFINITY
        if ( FPR_small == FPR_z )
            return FPR_big;

        FEGETENVD( FPR_env );                           // save environment, set default
        FESETENVD( FPR_z );

        temp = FPR_small / FPR_big;                     OldEnvironment.d = FPR_env;
        temp = sqrt ( FPR_one + temp * temp );

        FEGETENVD( CurrentEnvironment.d );
        CurrentEnvironment.i.lo &= ~FE_UNDERFLOW;       // Clear any inconsequential underflow
        FESETENVD( CurrentEnvironment.d );

        temp = FPR_big * temp;                          // Might raise UNDERFLOW or OVERFLOW

        FEGETENVD( CurrentEnvironment.d );
        OldEnvironment.i.lo |= CurrentEnvironment.i.lo; // Pick up any UF or OF
        FESETENVD( OldEnvironment.d );                  // restore caller's environment

        return temp;
}
```

Delivered In Jaguar: vecLib

- One stop shopping for math performance:
  - Digital Signal Processing,1D 2D FFTs [vDSP]
  - BLAS Levels 1,2,3 (ATLAS tuned, selectively SMP aware)
  - LAPACK, linear systems and eigenvalue problems
  - Tuned 4x4, 8x8, 16x16, 32x32 matrix multiplies
- Heavy use of AltiVec throughout
- Coding to vecLib in C:

```
#include <vecLib/vecLib.h>
cc someMath.c -O2 -framework vecLib
   Coming Soon ...
#include < Accelerate/Accelerate.h>
cc someMath.c -O2 -framework Accelerate
```

Aside: Can I Go Faster Than vecLib?

Aside: Can I Go Faster Than vecLib?

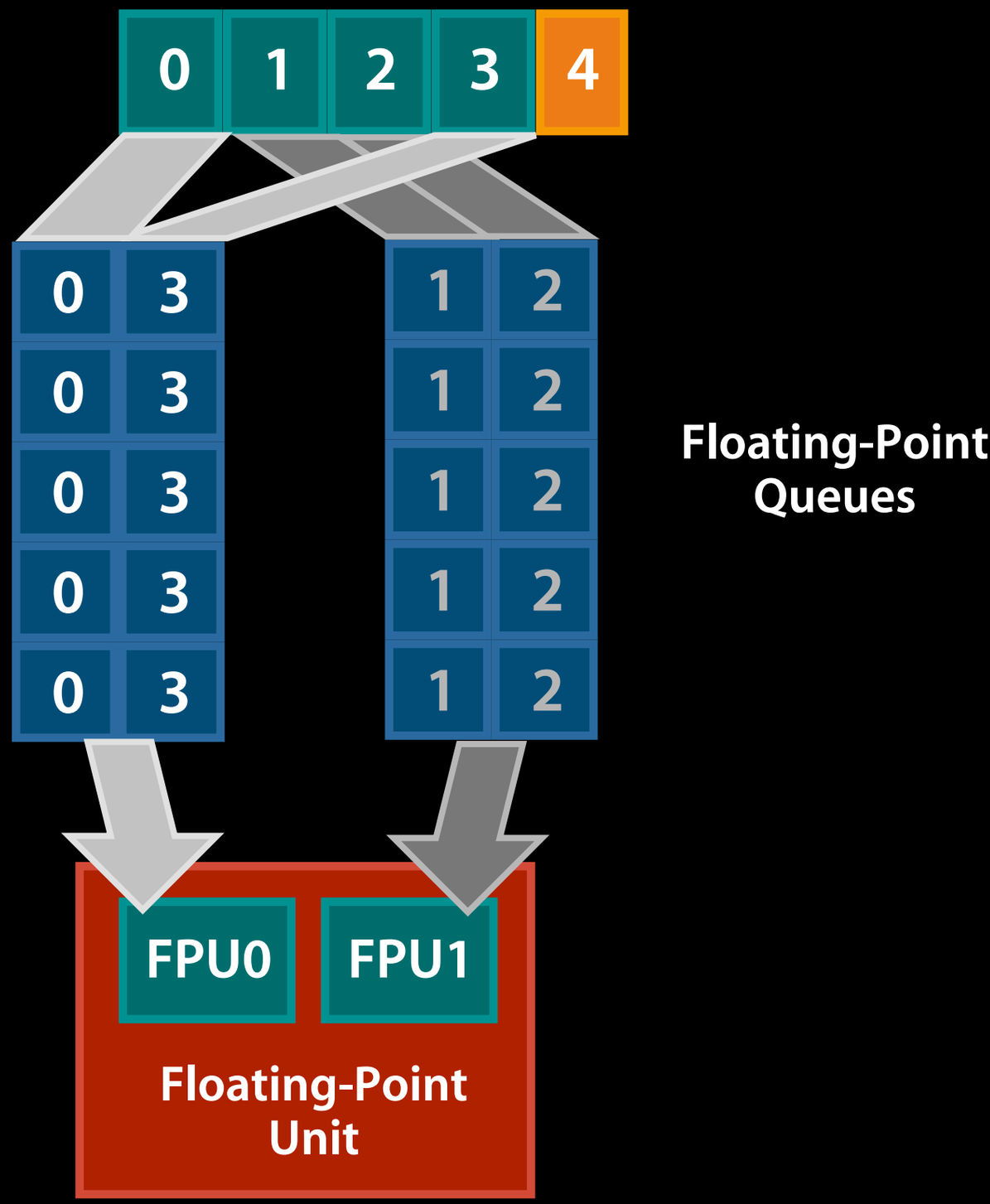
- Probably not

What Is New In Panther for Libm

- Tuned to 970:
  - Algorithms recast at instruction level to exploit 2 FPUs
  - Dual LSUs fetch data early and often
  - Careful attention paid to dispatch group formation
  - Careful attention paid to Load/Store hazards
  - sqrt is in hardware!
- Faster on G4 too!

Dispatch Group Formation

- Floating Point Issue:



Libm Performance: Cycle Counts

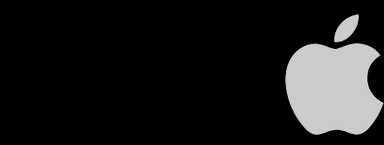
- Cycle count comparison (smaller count is faster)

Function	7455 cycles (Jaguar)	970 cycles	IA-32 x87 [Table C-6]*
sqrt	150	inline?(40/35 per FPU):(52)	38
cos	81	52	190 (throughput 130)
sin	84	53	160 (throughput 130)
tan	98	67	225 (throughput 130)
exp	127	88	180 (throughput 60)?
log	129	129	140 (throughput 85)
arcsin	128	121	188 (throughput 140)
arccos	122	115	188 (throughput 140)
arctan	106	97	150 (throughput 140)

*P4 Processor Optimization Reference Manual

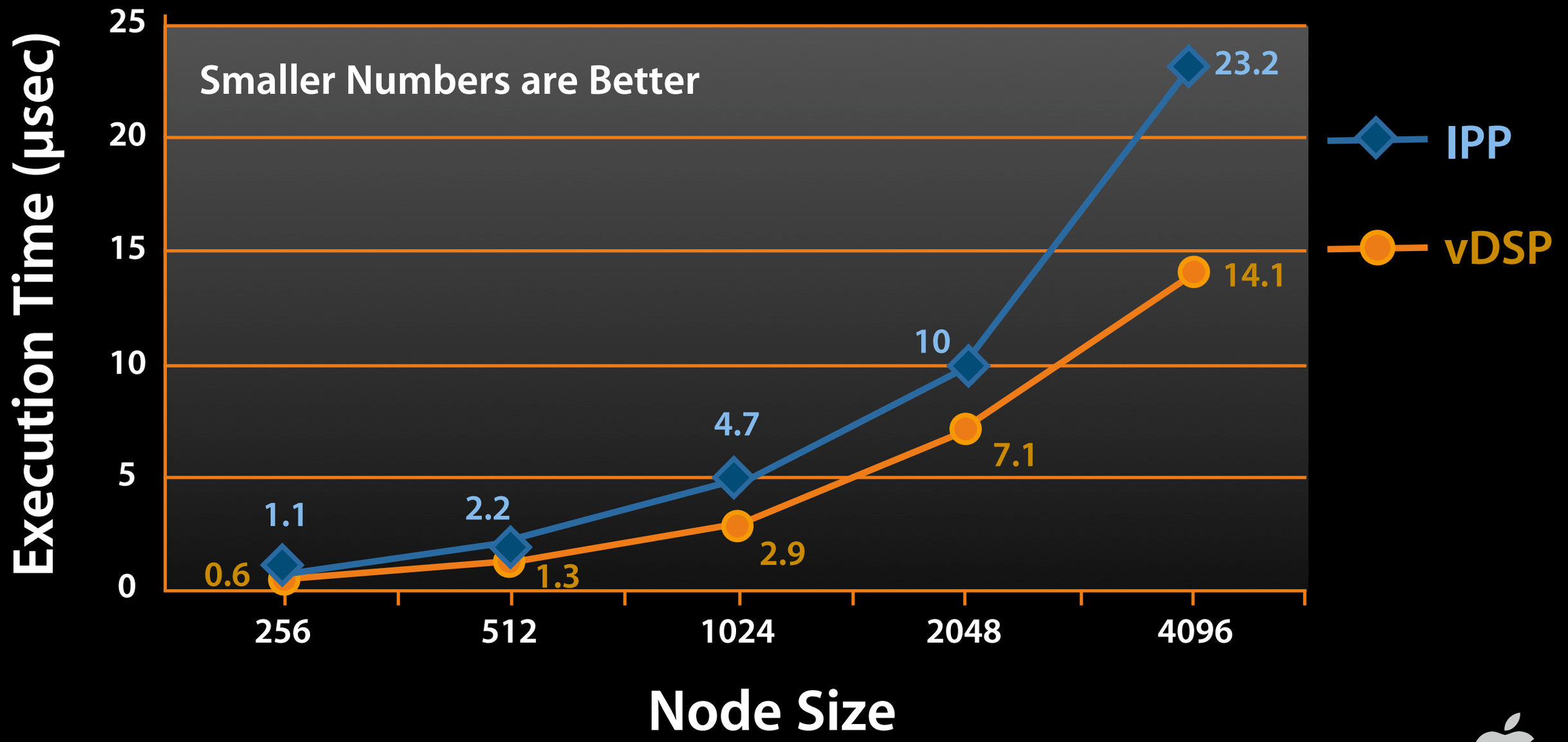
What Is New In Panther for vecLib

- DOUBLE PRECISION math performance:
  - DSP double precision routines tuned to 970
  - BLAS double precision routines tuned to 970:
    - matmul DOUBLE PRECISION kernel running in cache at 3.4flops per clock (84% peak, measured 6.7Gflops@2.0GHz)
    - DGEMM 1000x1000 running out of cache at 2.4flops per clock (60% peak, measured 4.8Gflops@2.0GHz)
    - DOUBLE PRECISION 4x4, 8x8, 16x16, 32x32 matrix multiplies tuned for 970
- Thread-safe, faster LAPACK



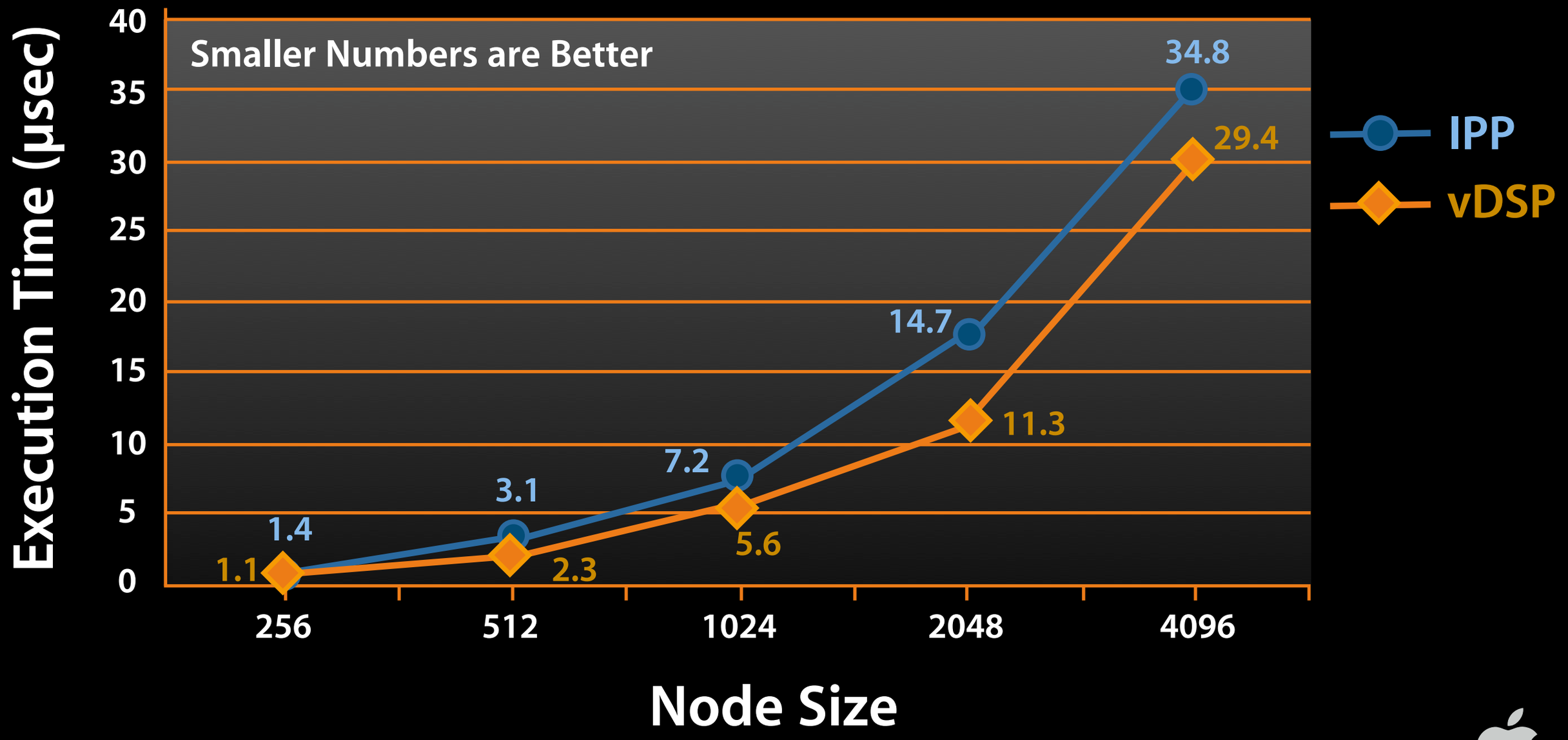
Performance of Real 1D-FFTs:

OS X vDSP on 2.0 GHz G5 vs. Linux IPP 3.0 GHz P4



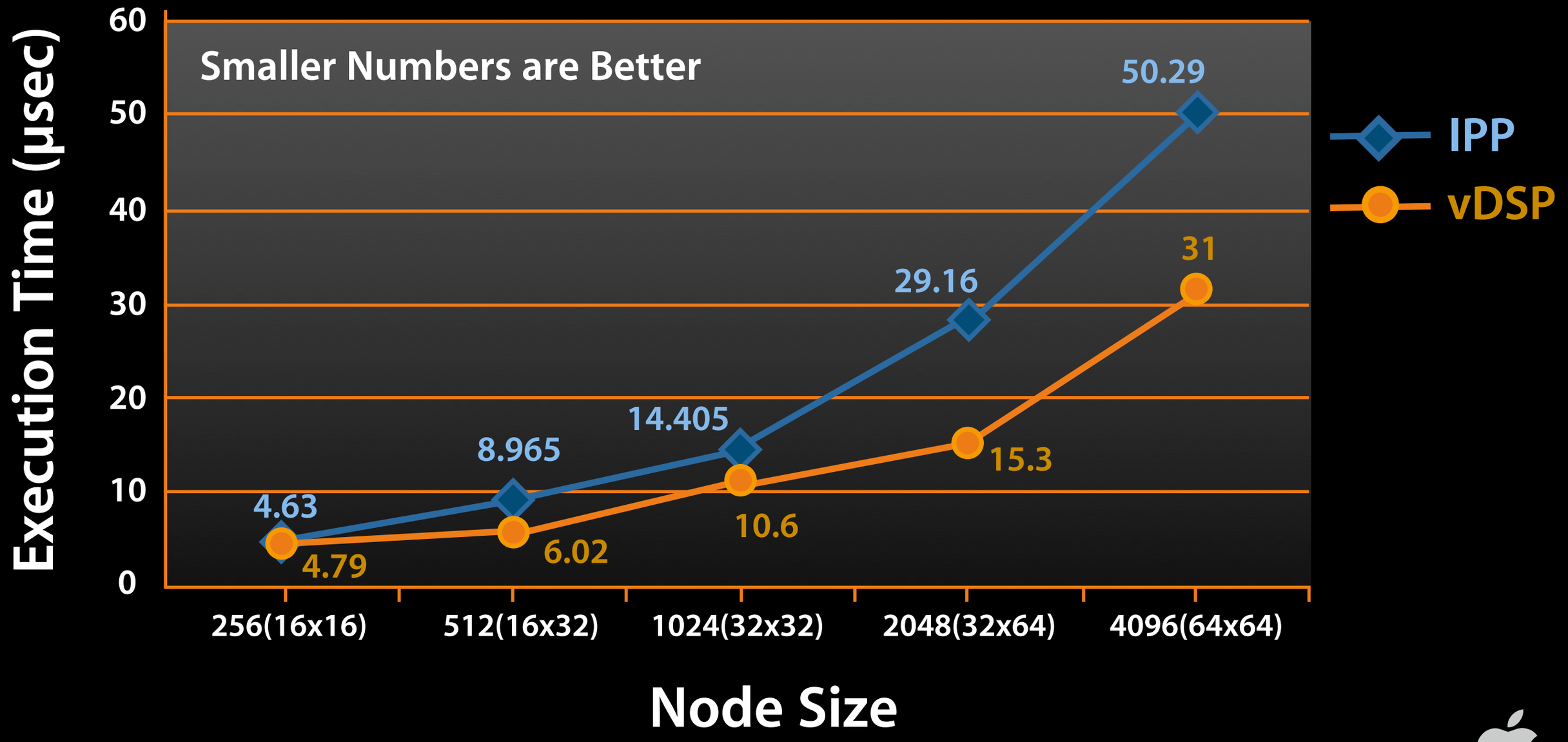
Performance of Complex 1D-FFTs:

OS X vDSP on 2.0 GHz G5 vs. Linux IPP 3.0 GHz P4



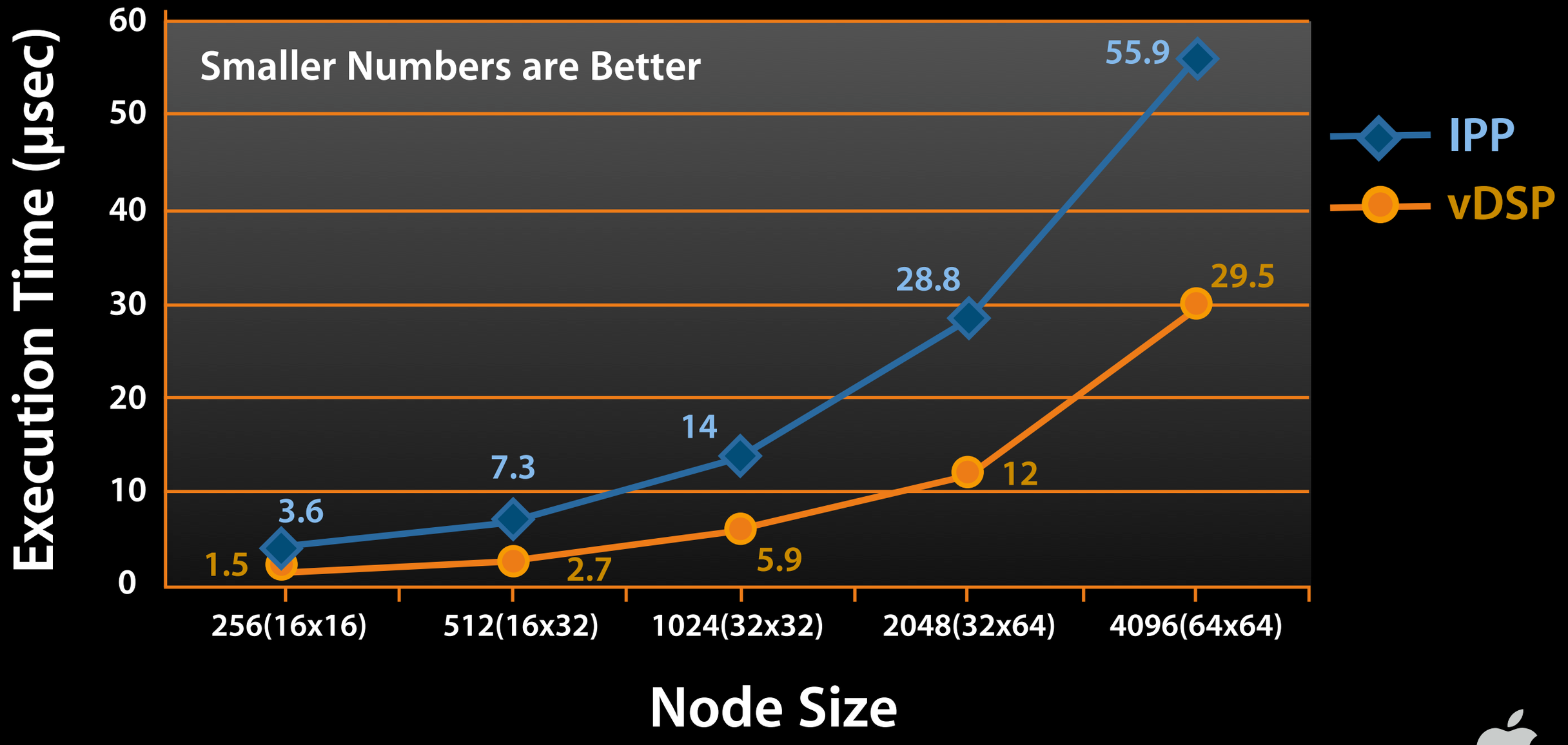
Performance of Real 2D-FFTs:

OS X vDSP on 2.0 GHz G5 vs. Linux IPP 3.0 GHz P4



Performance of Complex 2D-FFTs:

OS X vDSP on 2.0 GHz G5 vs. Linux IPP 3.0 GHz P4



vecLib Performance: LAPACK 1000

- LU Decomposition and solution of a 1000x1000 system of linear equations Ax=b.
- ATLAS based: blocked, SMP aware

Benchmark	2.0GHz PPC 970 (larger flops are better)
DOUBLE PRECISION, UniProcessor 2.0Ghz	dlp1000: 2.64Gflops
DOUBLE PRECISION, Dual Processor 2.0Ghz	dlp1000: 3.53Gflops
SINGLE PRECISION, UniProcessor 2.0Ghz	slp1000: 4.78Gflops
SINGLE PRECISION, Dual Processor 2.0Ghz	slp1000: 5.55Gflops

Accelerate: New Umbrella Framework

- Contents
  - Digital Signal Processing (vDSP)
  - Matrix and Vector computations (BLAS)
  - Linear Algebra Routines (LAPACK)
  - Vector libm (vMathLib)
  - Large Number arithmetic support (vBigNum)
  - Image Processing Library (vImage)

A Day In the Life: microOptimization

- First, PROFILE. Otherwise you will waste your time!
- Profiling on 970 often leads to “Aha!” discoveries
- “Sampler” command line tool gives you a coarse fix
- “Shark” zeroes in on the hot spots

CHUD Tools

Computer Hardware Understanding Developer Tools

Eric Miller

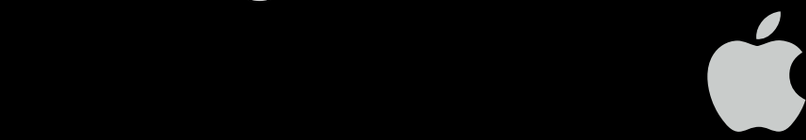
Senior Scientist—Architecture and Performance Group

Introduction to CHUD Tools

- Suite of performance analysis tools
- Low-level access to performance monitors
  - Processor
  - Memory controller
  - Operating system
- Helps you find problems and improve your code
- Freely available on the web and the Developer Tools CD!

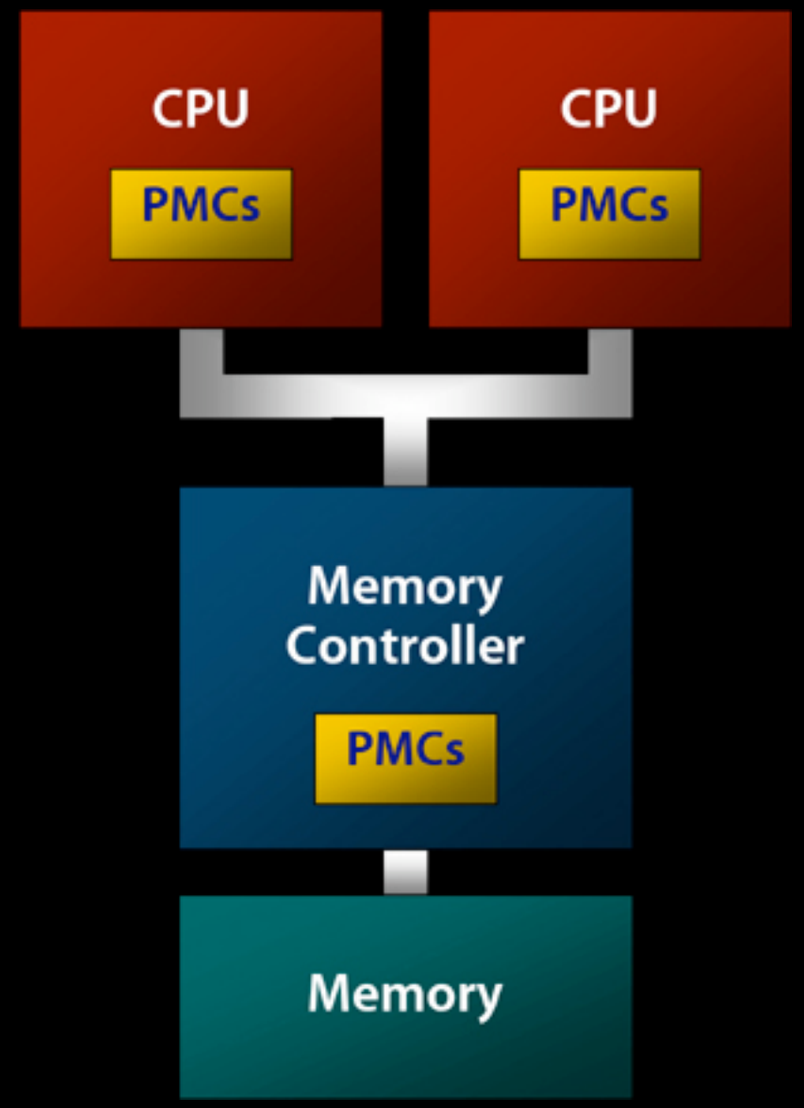
CHUD Tools 3.0

- Profiling
  - Shark—instruction-level profiler
  - MONster—spreadsheet for performance events
  - Saturn—function call graph visualizer
- Tracing
  - Amber—instruction tracer
  - Acid—trace analyzer
  - SimG4 and SimG5—cycle-accurate CPU simulators
- Instrumentation
  - CHUD.framework—API for performance tuning



Performance Counters

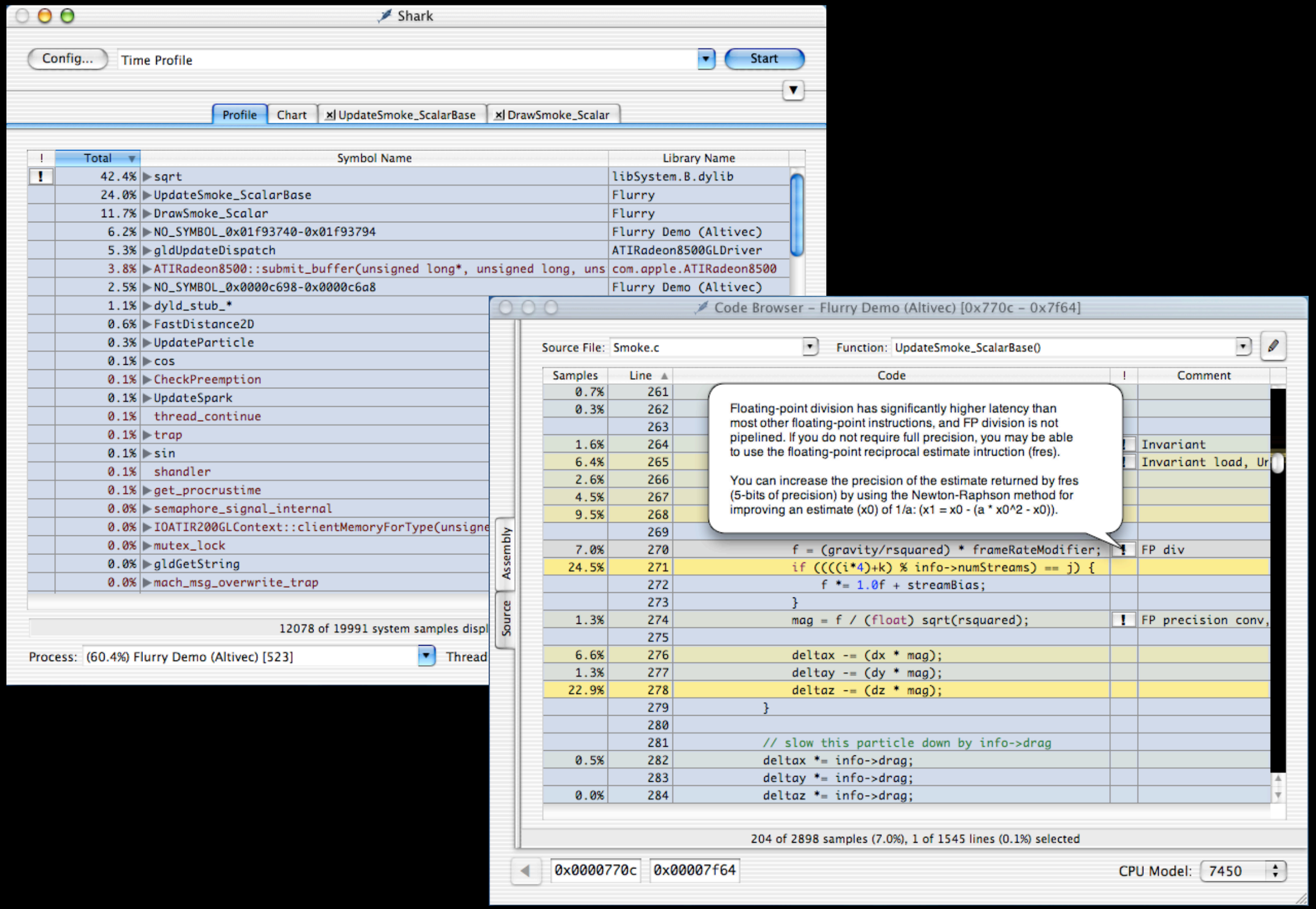
- Apple systems have performance monitor counters (PMCs)
- PMCs record performance "events"
  - Cache misses
  - Instructions completed
  - Execution stall cycles
  - Page faults
- Use CHUD Tools to configure, control and view the PMCs



Shark

- Profile a process, thread, or entire system
  - Time profile—visualize performance hot spots
  - Event profile—relate hardware events to source
  - Capture everything—drivers, kernel, applications
  - Very low overhead
- Automated Analysis
  - Annotated source code and disassembly
  - Static analysis to find sub-optimal code
  - Optimization tips
- Save and review sessions
- Command line version

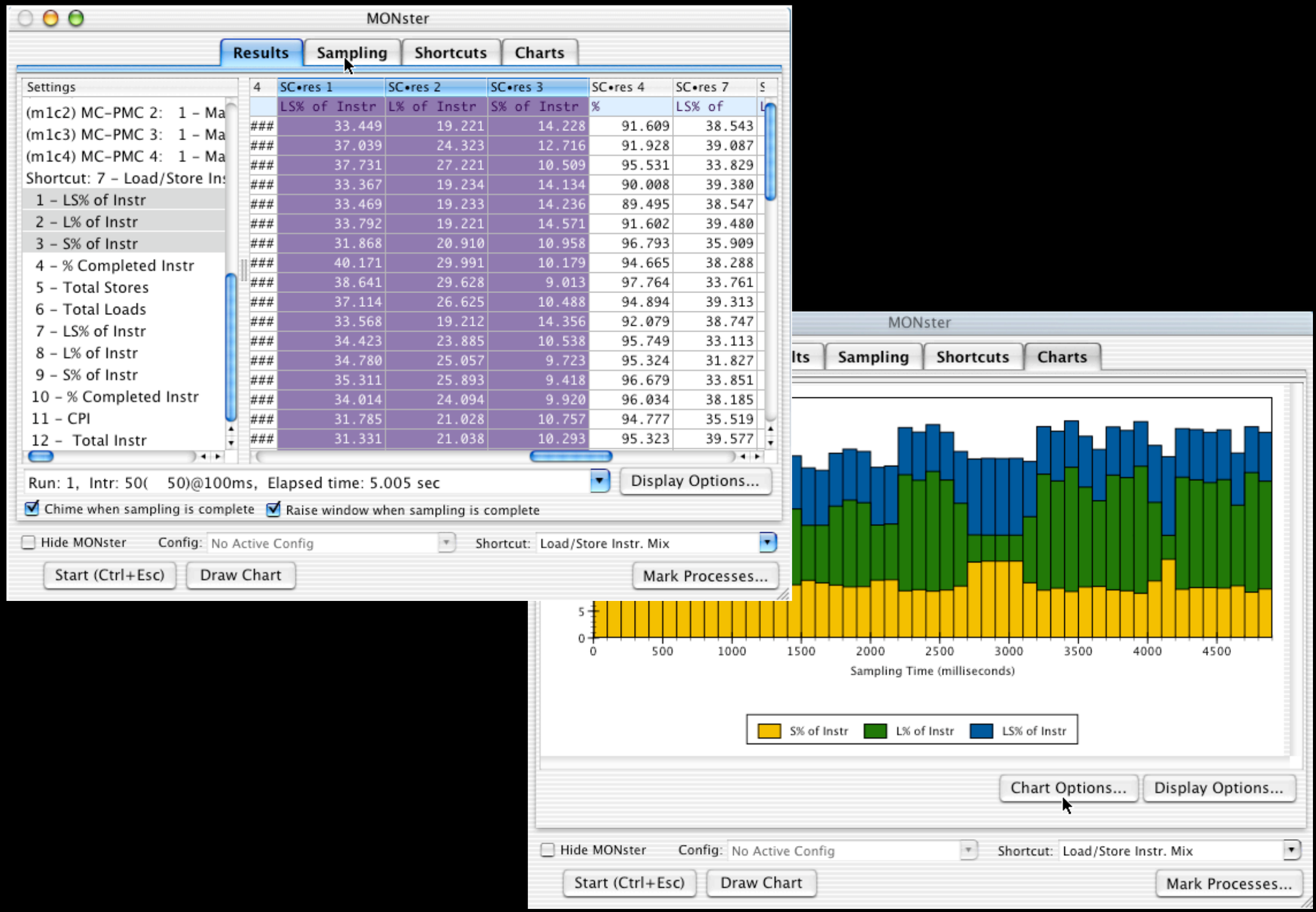
Shark


Shark
Config...
Time Profile
Start
Profile
Chart
UpdateSmoke_ScalarBase
DrawSmoke_Scalar
! Total Symbol Name Library Name
! 42.4% sqrt libSystem.B.dylib
24.0% UpdateSmoke_ScalarBase Flurry
11.7% DrawSmoke_Scalar Flurry
6.2% NO_SYMBOL_0x01f93740-0x01f93794 Flurry Demo (Altivec)
5.3% gldUpdateDispatch ATIRadeon8500GLDriver
3.8% ATIRadeon8500::submit_buffer(unsigned long*, unsigned long, uns com.apple.ATIRadeon8500
2.5% NO_SYMBOL_0x0000c698-0x0000c6a8 Flurry Demo (Altivec)
1.1% dyld_stub_*
0.6% FastDistance2D
0.3% UpdateParticle
0.1% cos
0.1% CheckPreemption
0.1% UpdateSpark
0.1% thread_continue
0.1% trap
0.1% sin
0.1% shandler
0.1% get_procrustime
0.0% semaphore_signal_internal
0.0% IOATIR200GLContext::clientMemoryForType(unsigne
0.0% mutex_lock
0.0% gldGetString
0.0% mach_msg_overwrite_trap
12078 of 19991 system samples displ
Process: (60.4%) Flurry Demo (Altivec) [523]
Thread
Code Browser – Flurry Demo (Altivec) [0x770c – 0x7f64]
Source File: Smoke.c
Function: UpdateSmoke_ScalarBase()
Samples Line Code ! Comment
0.7% 261
0.3% 262
263
1.6% 264 ! Invariant
6.4% 265 ! Invariant load, Ur
2.6% 266
4.5% 267
9.5% 268
269
7.0% 270 f = (gravity/rsquared) * frameRateModifier; ! FP div
24.5% 271 if ((((i*4)+k) % info->numStreams) == j) {
272 f *= 1.0f + streamBias;
273 }
1.3% 274 mag = f / (float) sqrt(rsquared); ! FP precision conv,
275
6.6% 276 deltax -= (dx * mag);
1.3% 277 deltay -= (dy * mag);
22.9% 278 deltaz -= (dz * mag);
279 }
280
281 // slow this particle down by info->drag
0.5% 282 deltax *= info->drag;
283 deltay *= info->drag;
0.0% 284 deltaz *= info->drag;
Floating-point division has significantly higher latency than most other floating-point instructions, and FP division is not pipelined. If you do not require full precision, you may be able to use the floating-point reciprocal estimate intruction (fres).
You can increase the precision of the estimate returned by fres (5-bits of precision) by using the Newton-Raphson method for improving an estimate (x0) of 1/a: (x1 = x0 - (a * x0^2 - x0)).
Assembly
Source
204 of 2898 samples (7.0%), 1 of 1545 lines (0.1%) selected
0x0000770c
0x00007f64
CPU Model: 7450

MONster

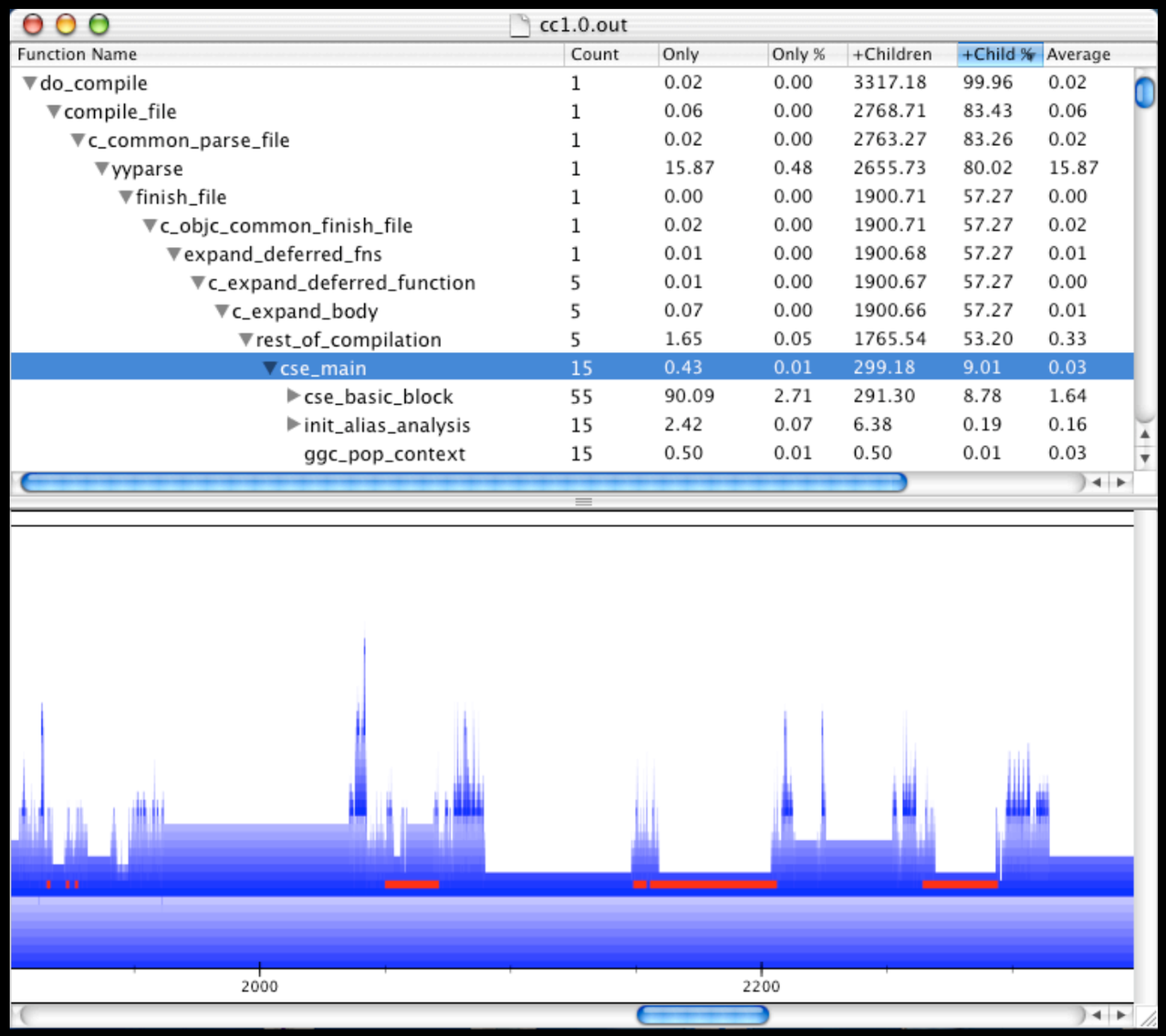
- Configure Performance Monitor Counters (PMCs)
- Collect PMC data based on:
  - Timed intervals, event counts or hotkey
- Shortcuts—compute performance metrics:
  - Memory bandwidth, cycles per instruction
  - Define your own with infix expressions
- View results in tabular or chart form
- Save and review sessions
- Command line version

MONster

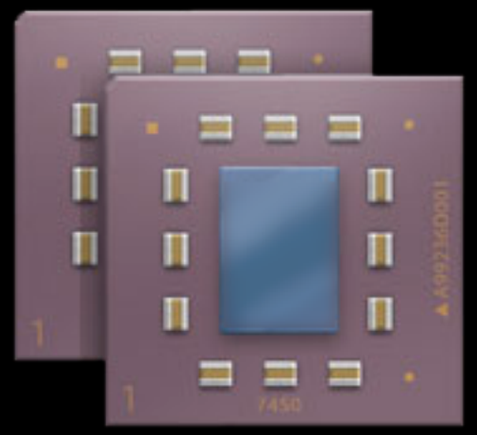

MONster
Results
Sampling
Shortcuts
Charts
Settings
(m1c2) MC-PMC 2: 1 - Ma
(m1c3) MC-PMC 3: 1 - Ma
(m1c4) MC-PMC 4: 1 - Ma
Shortcut: 7 - Load/Store In
1 - LS% of Instr
2 - L% of Instr
3 - S% of Instr
4 - % Completed Instr
5 - Total Stores
6 - Total Loads
7 - LS% of Instr
8 - L% of Instr
9 - S% of Instr
10 - % Completed Instr
11 - CPI
12 - Total Instr
4 | SC•res 1 | SC•res 2 | SC•res 3 | SC•res 4 | SC•res 7
LS% of Instr | L% of Instr | S% of Instr | % | LS% of
| 33.449 | 19.221 | 14.228 | 91.609 | 38.543
| 37.039 | 24.323 | 12.716 | 91.928 | 39.087
| 37.731 | 27.221 | 10.509 | 95.531 | 33.829
| 33.367 | 19.234 | 14.134 | 90.008 | 39.380
| 33.469 | 19.233 | 14.236 | 89.495 | 38.547
| 33.792 | 19.221 | 14.571 | 91.602 | 39.480
| 31.868 | 20.910 | 10.958 | 96.793 | 35.909
| 40.171 | 29.991 | 10.179 | 94.665 | 38.288
| 38.641 | 29.628 | 9.013 | 97.764 | 33.761
| 37.114 | 26.625 | 10.488 | 94.894 | 39.313
| 33.568 | 19.212 | 14.356 | 92.079 | 38.747
| 34.423 | 23.885 | 10.538 | 95.749 | 33.113
| 34.780 | 25.057 | 9.723 | 95.324 | 31.827
| 35.311 | 25.893 | 9.418 | 96.679 | 33.851
| 34.014 | 24.094 | 9.920 | 96.034 | 38.185
| 31.785 | 21.028 | 10.757 | 94.777 | 35.519
| 31.331 | 21.038 | 10.293 | 95.323 | 39.577
Run: 1, Intr: 50(50)@100ms, Elapsed time: 5.005 sec
Display Options...
Chime when sampling is complete
Raise window when sampling is complete
Hide MONster
Config: No Active Config
Shortcut: Load/Store Instr. Mix
Start (Ctrl+Esc)
Draw Chart
Mark Processes...
MONster
lts
Sampling
Shortcuts
Charts
5
0
0
500
1000
1500
2000
2500
3000
3500
4000
4500
Sampling Time (milliseconds)
S% of Instr
L% of Instr
LS% of Instr
Chart Options...
Display Options...
Hide MONster
Config: No Active Config
Shortcut: Load/Store Instr. Mix
Start (Ctrl+Esc)
Draw Chart
Mark Processes...

Saturn

- Record function call history
- Instruments functions at entry/exit with GCC
- Visualize call tree
- For each function:
  - Call counts
  - PMC event counts
  - Execution times



Instruction Tracing and Simulation



- Amber—instruction tracer
- Acid—trace analyzer
  - Trace statistics
  - Memory footprint (gnuplot output)
  - Points out problematic instruction sequences
    - Dependency stalls, spilled loads, unaligned operands
- SimG4–PPC7400 cycle-accurate simulator
- SimG5–PPC970 cycle-accurate simulator (coming soon)

CHUD.framework

- Instrument your source code
  - Start and stop MONster or Shark
  - Directly read and report on PMCs
- Write your own performance analysis tools
  - Control Performance Monitor Counters
  - Collect information about system hardware
    - CPU, memory controller, bus
- Extensive HTML reference guide

CHUD.framework Example #1

```
#include <CHUD/chud.h>
…
chudInitialize();

// MONster or Shark is in remote mode
chudAcquireRemoteAccess();

chudStartRemotePerfMonitor(“run_01”);

your_important_function();

chudStopRemotePerfMonitor();

chudReleaseRemoteAccess();
```

CHUD.framework Example #2

```
chudInitialize();
chudAcquireSamplingFacility();

chudSetMarkFilter(chudAllCPUDev, chudFmodeOnlyMarked);
chudMarkPID(getpid(), TRUE);


chudSetCounterEvent(chudAllCPUDev, PMC_1, 1);
chudSetCounterEvent(chudAllCPUDev, PMC_2, 2);

chudClearCounters();
chudStartCounters();

important_function();

chudStopCounters();

do_something_with_results();
chudReleaseSamplingFacility();
```

For More Information

- On the Web:
  http://developer.apple.com/tools/debuggers.html
- Send Raves, Rants, and Requests:
  chud-tools-feedback@group.apple.com

Optimization Case Study

Steve Peters

microOptimization: Memory

- Load data to registers early, well in advance of its use
- Load data often, harness 2 LSUs and 2 FPUs
- Load data sequentially, hardware initiated prefetch streams are very effective
- dst{t} and vec_dst{t} are bad eggs, they force execution synchronization on the 970. Re-think these as dcbtl
- dcb{z,t} are OK, but note that 970 cache line size = 128b

microOptimization: FPUs

- Each 970 core has TWO IEEE-754 compliant, DOUBLE PRECISION floating point units
- At each cycle, on each FPU, fmadd class instructions develop 2 flops: Net 4 flops per cycle, and is achievable
- 32 floating point registers and 48 renames
- Typical latency: 6 cycles. Throughput: 1. (key exceptions are fsqrt, fdiv.)
- Coding “as if” a 12 cycle pipe (when possible) generally results in full FPUx2 utilization

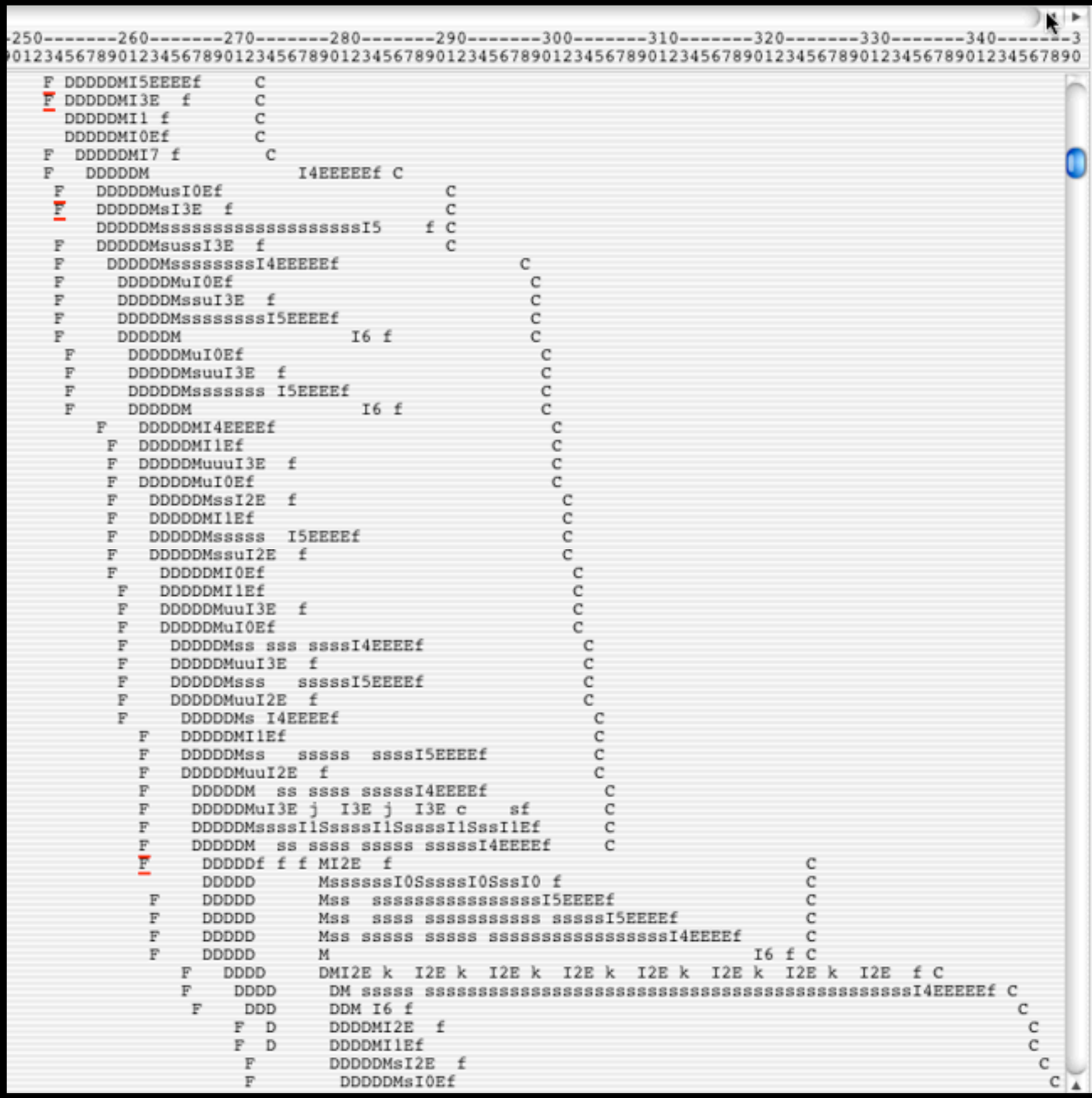
macroOptimization: Algorithm Choice

- Rethinking an algorithm to make better use of the new architecture
- Example: it takes two hands to matmul C = A • B ...
- (Junior) High School—product of NxN matrices: dot product of length N repeated N-squared times. Computing each Cij requires 2N fetches.
- vecLib dgemm—accumulate 16 inner products simultaneously. Computing 16 Cij's requires 8N fetches, that's a factor of 4 reduction in memory bandwidth.

Case Study: sin() Tuning for 970 Libm

```
absOfX = __FABS ( x );

FEGETENVD( OldEnvironment.d );                  // save env, set default
FESETENVD( 0.0 );

if ( absOfX < piOver4 ) {                       // |x| < π/4
      if ( absOfX == 0.0) {
              FESETENVD( OldEnvironment.d );    // restore caller's mode
              return x;                         // +0 -0 preserved
      }

      xSquared = x * x;                         // sin polynomial approximation
      xFourth = xSquared * xSquared;
      OldEnvironment.i.lo |= FE_INEXACT;
      temp1 = s9 + s13*xFourth;
      temp2 = s7 + s11*xFourth;
      temp1 = s5 + temp1*xFourth;
      temp2 = s3 + temp2*xFourth;
      xThird = xSquared * x;
      temp1 = temp2 + xSquared * temp1;
      result = x + xThird * temp1;

      if ( fabs ( result ) < kMinNormal )
              OldEnvironment.i.lo |= FE_UNDERFLOW;

      FESETENVD( OldEnvironment.d );            // restore caller's mode
      return ( result ) ;
}
```

1030: 9003fa30	fabs	F12,F1	1204	...IIDEEFR.....	1210
1031: 9003fa34	stwu	R1,0xff90(R1)	1204	...IIDEEEER.....	1211
1032: 9003fa38	mfspr	R31,LR	1205	....IIDEEEER.....	1212
1033: 9003fa3c	mffs	F0	1205	....IIIIIIIIDEEFR.....	1217
1034: 9003fa40	addis	R3,R31,0xb	1206	.....IIIIIIIIIIIDR.....	1219
1035: 9003fa44	stfd	F0,0x50(R1)	1206	.....IIIIIIIIIIIDER.....	1220
1036: 9003fa48	lfd	F2,0x85dc(R3)	1207	......IIIIIIIIIIIDER.....	1221
1037: 9003fa4c	mtfsf	0xf,F2	1207	......IIIIIIIIIIIDDDEEFR.....	1225
1038: 9003fa50	addis	R2,R31,0xb	1208	.......IIIIIIIIIIIIIIIIIDR.....	1227
1039: 9003fa54	lfd	F2,0x8674(R2)	1214	.............IIIIIIIIIIIDDER.....	1229
1040: 9003fa58	fcmpu	0,F12,F2	1219	..................IIIIIIIIDDDEEFR.....	1233
1041: 9003fa5c	bc-	4,0,0xa8	1219	..................IIIIIIII@@@@@@@.....	1233
1042: 9003fa60	addis	R5,R31,0xb	1220	...................IIIIIIIIDFFFFFR.....	1234
1043: 9003fa64	lfd	F3,0x85dc(R5)	1220	...................IIIIIIIIDDEFFFR.....	1234
1044: 9003fa68	fcmpu	0,F12,F3	1227	............................IIIDDEEFR.....	1235
1045: 9003fa6c	bc-	4,2,0x10	1227	............................@@@@@@@@@.....	1235
1046: 9003fa7c	fmul	F3,F1,F1	1228	..............................IIIIIIDEEFR.....	1238
1047: 9003fa80	addis	R11,R31,0xb	1229	..............................IIIIIDFFFR.....	1238
1048: 9003fa84	lfd	F0,0x86b4(R11)	1229	..............................IIIIIIDEFFR.....	1239
1049: 9003fa88	addis	R2,R31,0xb	1229	..............................IIIIIIDFFFR.....	1239
1050: 9003fa8c	lfd	F7,0x86ac(R2)	1229	..............................IIIIIIIDEFFR.....	1240
1051: 9003fa90	addis	R12,R31,0xb	1231	..............................IIIIIDFFFR.....	1240
1052: 9003fa94	fmul	F4,F3,F3	1235	..................................IIDEEFR.....	1241
1053: 9003fa98	lfd	F6,0x86bc(R12)	1235	..................................IIDEFFR.....	1241
1054: 9003fa9c	addis	R9,R31,0xb	1236	...................................IIIDFFR.....	1242
1055: 9003faa0	addis	R10,R31,0xb	1237	...................................IIDFFR.....	1242
1056: 9003faa4	lfd	F5,0x86c4(R9)	1237	...................................IIIDEFR.....	1243
1057: 9003faa8	addis	R8,R31,0xb	1237	...................................IIIDFFR.....	1243
1058: 9003faac	fmadd	F10,F4,F7,F6	1238	...................................IIIDEEFR.....	1245
1059: 9003fab0	lfd	F12,0x86cc(R10)	1238	...................................IIIDEFFR.....	1245
1060: 9003fab4	fmadd	F9,F4,F0,F5	1240	....................................IIDEEFR.....	1246
1061: 9003fab8	lfd	F11,0x86d4(R8)	1240	....................................IIDEFFR.....	1246
1062: 9003fabc	fmul	F8,F3,F1	1241	....................................IIDEEFR.....	1247
1063: 9003fac0	addis	R6,R31,0xb	1242	....................................IDFFFR.....	1247
1064: 9003fac4	fmadd	F13,F10,F4,F12	1242	....................................IIIDEEFR.....	1249
1065: 9003fac8	lfd	F5,0x8694(R6)	1242	....................................IIIIDEFR.....	1249
1066: 9003facc	fmadd	F2,F9,F4,F11	1243	....................................IIIDEEFR.....	1250
1067: 9003fad0	lwz	R11,0x54(R1)	1243	....................................IIIIDEFR.....	1250
1068: 9003fad4	oris	R7,R11,0x200	1244	....................................IIIDDDFR.....	1251
1069: 9003fad8	fmadd	F7,F3,F13,F2	1244	....................................IIIIDDEEFR.....	1253
1070: 9003fadc	stw	R7,0x54(R1)	1246	....................................IIDEEEEER.....	1254
1071: 9003fae0	fmadd	F1,F8,F7,F1	1247	....................................IIIDDDEEFR.....	1256
1072: 9003fae4	fabs	F6,F1	1247	....................................IIIIIIDDDEEFR.....	1259
1073: 9003fae8	fcmpu	0,F6,F5	1248	....................................IIIIIIIIDDDEEFR.....	1262
1074: 9003faec	bc-	4,0,0xc	1248	....................................@@@@@@@@@@@@@@.....	1262
1075: 9003faf8	lfd	F8,0x50(R1)	1249	....................................IIIIIIIDDDEFFFR.....	1263
1076: 9003fafc	mtfsf	0xf,F8	1249	....................................IIIIIIIIIIDDEEFR.....	1264
1077: 9003fb00	b	0x278	1250	....................................@@@@@@@@@@@@@@.....	1250
1078: 9003fd78	lwz	R9,0x78(R1)	1251	....................................IIIIIIIIIIIIIDER.....	1267
1079: 9003fd7c	addi	R1,R1,0x70	1251	....................................IIIIIIIIIIIIIDFR.....	1267
1080: 9003fd80	lwz	R31,0xfffc(R1)	1254	....................................IIIIIIIIIIIDER.....	1268
1081: 9003fd84	mtspr	LR,R9	1257	....................................IIIIIIIIIDDER.....	1269
1082: 9003fd88	bclr+	20,0	1257	....................................@@@@@@@@@@@@@.....	1268



Inst Addr	Mnemonic
9003fa30	fabs F12,F1
9003fa34	stwu R1,1,(R1)
9003fa34	stwu R1,1,(R1)
9003fa34	stwu R1,1,(R1)
9003fa38	mfspr R31,LR
9003fa3c	mffs F0
9003fa40	addis R3,R31,11
9003fa44	stfd F0,0,(R1)
9003fa44	stfd F0,0,(R1)
9003fa48	lfd F2,2,(R3)
9003fa4c	mtfsf 0x,ff,F2
9003fa50	addis R2,R31,11
9003fa54	lfd F2,2,(R2)
9003fa58	fcmpu F0,F12,F2
9003fa5c	bc 4,0,.+168
9003fa60	addis R5,R31,11
9003fa64	lfd F3,3,(R5)
9003fa68	fcmpu F0,F12,F3
9003fa6c	bc 4,2,.+16
9003fa7c	fmul F3,F1,F1
9003fa80	addis R11,R31,11
9003fa84	lfd F0,0,(R11)
9003fa88	addis R2,R31,11
9003fa8c	lfd F7,7,(R2)
9003fa90	addis R12,R31,11
9003fa94	fmul F4,F3,F3
9003fa98	lfd F6,6,(R12)
9003fa9c	addis R9,R31,11
9003faa0	addis R10,R31,11
9003faa4	lfd F5,5,(R9)
9003faa8	addis R8,R31,11
9003faac	fmadd F10,F4,F7,F6
9003fab0	lfd F12,12,(R10)
9003fab4	fmadd F9,F4,F0,F5
9003fab8	lfd F11,11,(R8)
9003fabc	fmul F8,F3,F1
9003fac0	addis R6,R31,11
9003fac4	fmadd F13,F10,F4,F12
9003fac8	lfd F5,5,(R6)
9003facc	fmadd F2,F9,F4,F11
9003fad0	lwz R11,11,(R1)
9003fad4	oris R7,R11,0x512
9003fad8	fmadd F7,F3,F13,F2
9003fadc	stw R7,7,(R1)
9003fadc	stw R7,7,(R1)
9003fae0	fmadd F1,F8,F7,F1
9003fae4	fabs F6,F1
9003fae8	fcmpu F0,F6,F5
9003faec	bc 4,0,.+12
9003faf8	lfd F8,8,(R1)
9003fafc	mtfsf 0x,ff,F8
9003fb00	b .+632
9003fd78	lwz R9,9,(R1)
9003fd7c	addi R1,R1,112
9003fd80	lwz R31,31,(R1)
9003fd84	mtspr LR,R9

```
-180-------190-------200-------210-------220-------230-------240-------250-------260-------270-------280-
90123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123
 F    DDDD  DMsssI4EEEEf                                                C
 F    DDDD  DMss ssssI5EEEEf                                            C
  F    DDDD  DMss sssI4EEEEEf                                                C
  F     DDD  DDMs sss  I4EEEEf                                                C
  F     DDD  DDM                I6 f                                          C
  F      DD  DDDf f f M I4EEEEf                                                C
  F      DD  DDD      M         I6 f                                          C
  F       D  DDD      Df MI4EEEEf                                                C
  F       D  DDD      D  MI1Ef                                                  C
  F       D  DDD      D  MssI3E  f                                              C
   F      D  DDD      D  M s ssssI4EEEEf                                        C
   F         DDDD        DMI0Ef                                                  C
   F         DDDD        DMsuI3E  f                                              C
   F         DDDD        DMI1Ef                                                  C
   F         DDDD        DMssI2E  f                                              C
   F          DDD      D  DMI0Ef                                                  C
   F          DDD      D  DMuuI3E  f                                              C
   F          DDD      D  DMI1Ef                                                  C
    F         DDD      D  DMsuI2E  f                                              C
    F          DD      D  DDc c M I4EEEEf                                          C
    F          DD      D  DD    Ms  I5EEEEf                                        C
    F          DD      D  DD    MI1Ef                                              C
    F          DD      D  DD    MssI2E  f                                          C
    F           D      D  DD    Dc c c MI0Ef                                        C
    F           D      D  DD    D      MssI3E  f                                    C
    F           D      D  DD    D      MsI5EEEEf                                    C
     F          D      D  DD    D      MsI4EEEEf                                    C
     F                 D  DDD          Dc c MssI4EEEEf                                C
     F                 D  DDD          D    MssI5EEEEf                                C
     F                 D  DDD          D    Mss sssssI5EEEEf                          C
     F                 D  DDD          D    Mss sssssssssssI4EEEEf                    C
     F                    DDD    D          Dc c c c Msssss ss sssssI4EEEEEf                C
     F                     DD    D      D            DMI4EEEEf                              C
     F                     DD    D      D            DMssssssssssI5EEEEf                    C
       F                   DD    D      D            DMssssssssss sssss I5EEEEf              C
       F                    D    D      D    D        DMssssssssssssssssssssssssI7 f           C
                            D    D      D    D        DMssssssssssssssssssssssss sI7 f         C
       F                    D    D      D    D        DM                          I6 f        C
                F              D      D    D        DDMI2E c    sf                            C
                F              D      D    D        DDMI1Ef                                  C
                F                     D    D        DDDMssI0SssI0Ef                            C
                F                     D    D        DDDMsI3E c    sf                           C
                F                     D    D        DDDM          I6 f                         C
```

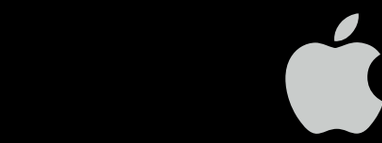
Inst Addr	Mnemonic	
25f8	fabs	F11,F1
2600	mtfsf	0x,ff,F12
2604	fcmpu	F0,F11,F2
2608	bc	4,0,.+148
260c	fcmpu	F0,F11,F12
2610	bc	12,2,.+132
2614	fmul	F7,F1,F1
2618	addis	R12,R31,0
261c	lfd	F5,5,(R12)
2620	fmadd	F13,F12,F12,F5
2624	addis	R11,R31,0
2628	lfd	F3,3,(R11)
262c	addis	R10,R31,0
2630	lfd	F9,9,(R10)
2634	addis	R8,R31,0
2638	lfd	F11,11,(R8)
263c	addis	R7,R31,0
2640	lfd	F6,6,(R7)
2644	fmul	F8,F7,F7
2648	fmul	F2,F1,F7
264c	addis	R6,R31,0
2650	lfd	F12,12,(R6)
2654	addis	R5,R31,0
2658	lfd	F13,13,(R5)
265c	fmadd	F10,F9,F8,F3
2660	fmadd	F0,F6,F8,F11
2664	fmadd	F6,F10,F8,F12
2668	fmadd	F9,F0,F8,F13
266c	fmadd	F0,F6,F7,F9
2670	fmadd	F1,F0,F2,F1
2674	mtfsf	0x,ff,F4
2678	fmul	F0,F3,F3
267c	fabs	F2,F1
2680	fcmpu	F4,F2,F5
2684	??-19	0x4c453382
2684	??-19	0x4c453382
2688	bc	12,2,.+672
2928	lwz	R9,9,(R1)
292c	addi	R1,R1,112
2930	mtspr	LR,R9
2934	lwz	R31,31,(R1)
2938	blr	

Case Study: sin() Tuning for 970 Libm

```
FEGETENVD( FPR_env );                                                   // save env, set default
FPR_z = 0.0f;
FPR_pi4 = piOver4;
absOfX = __FABS ( x );                          __ENSURE( FPR_z, FPR_z, FPR_pi4 );// ensure z and pi4 are in registers

FESETENVD( FPR_z );
if ( absOfX < FPR_pi4 ) {                                               // |x| < π/4
      if ( absOfX != FPR_z ) {
          register double FPR_s3, FPR_s5, FPR_s7, FPR_s9, FPR_s11, FPR_s13;

          xSquared = __FMUL( x, x );
          FPR_Min = kMinNormal;
          __ENSURE( FPR_z, FPR_z, FPR_Min );                            // ensure FPR_Min is loaded
          FPR_s9 = s9;                          FPR_s13 = s13;
          FPR_s7 = s7;                          FPR_s11 = s11;

          xFourth = __FMUL( xSquared, xSquared );   xThird = __FMUL( x, xSquared );
          FPR_s5 = s5;
          FPR_s3 = s3;
          temp1 = __FMADD( s13, xFourth, s9 );      temp2 = __FMADD( s11, xFourth, s7 );

          temp1 = __FMADD( temp1, xFourth, s5 );    temp2 = __FMADD( temp2, xFourth, s3 );

          temp1 = __FMADD( temp1, xSquared, temp2 );result = __FMADD( temp1, xThird, x );

          FESETENVD( FPR_env );                                         // restore caller's mode

          if ( __FABS( result ) < FPR_Min )
              __PROG_UF_INEXACT( FPR_Min );
          else
              __PROG_INEXACT( FPR_pi4 );

          return ( result );
      }
      else {
          FESETENVD( FPR_env );                                         // restore caller's mode
          return x;                                                     // +0 -0 preserved
      }
}
```

Summary of the Regimen

- CHUD
- Attention to load/store issues
- Dual (FPU) thinking
- Use all the registers
- Let the core do the walking through memory
- Dispatch group, dispatch slot to execution unit “rails”



Wrap Up

Roadmap

103 Image: Accelerating Image Processing	Mission Wed., 9:00am
506 CHUD: Performance Optimization Tools in Depth	Marina Wed., 2:00pm
305 Tuning Software with Performance Tools	Presidio Wed., 5:00pm

Who to Contact

Steve Peters
Senior Software Scientist
scp@apple.com

Ali Sazegari
Engineering Manager, CoreOS
sazegari@apple.com

http://developer.apple.com/wwdc2003/urls.html

For More Information

Reference Library

Technical Notes

TN2086: Tuning for G5: A Practical Guide
http://developer.apple.com/technotes/tn/tn2086.html

TN2087: Quick Comparison: G4 & G5
http://developer.apple.com/technotes/tn/tn2087.html

Reference Library

Documentation

Accelerate
vecLib
http://developer.apple.com/hardware/ve/vector_libraries.html

vImage
ADC Member Site > Download Software >

Velocity Engine
http://developer.apple.com/hardware/ve/index.html

On the Internet

“Inside the PowerPC 970, pts. I and II”

http://www.arstechnica.com/

Q&A

Mark Tozer-Vilchez
Desktop Hardware Evangelist
tozer@apple.com

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference